Tiled QR Decomposition and Its Optimization on CPU and GPU Computing System

Dongjin Kim and Kyu-Ho Park

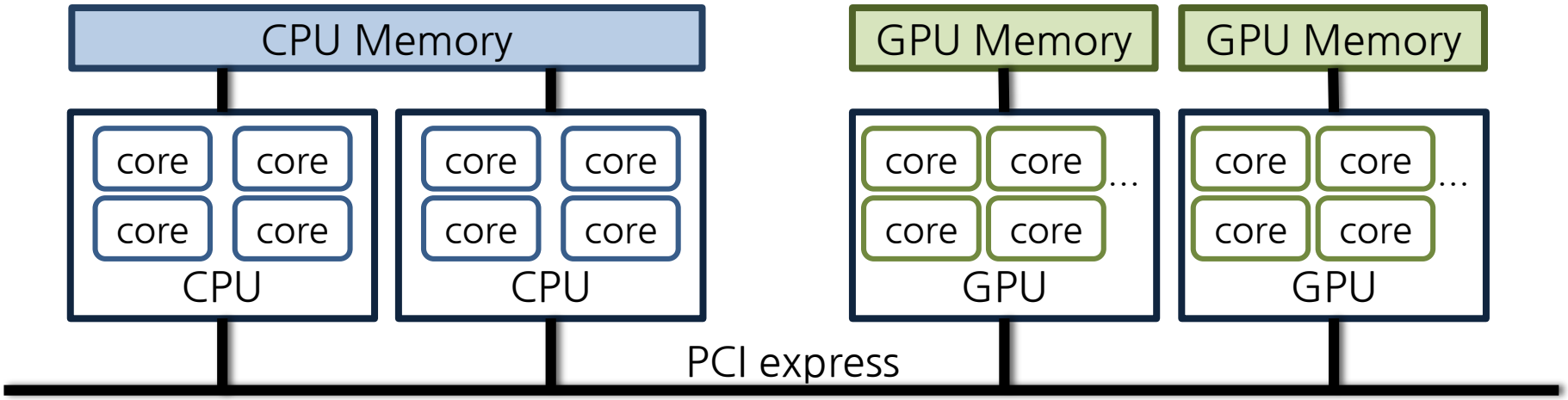
Presentation by Dongjin Kim

Ph.D. Student, CORE lab., Electrical Engineering, KAIST
djkim@core.kaist.ac.kr
October 1st, 2013
@ P2S2-2013

Contents

Heterogeneous Core System

- Common to use **heterogeneous cores** for performance
  - As distributed-memory system
- Properties
  - ① Performance heterogeneity
    - Different computation speed
  - ② Explicit memory copy needed
  - ③ GPGPUs expect a larger input than CPUs
    - Much more parallel cores than CPU



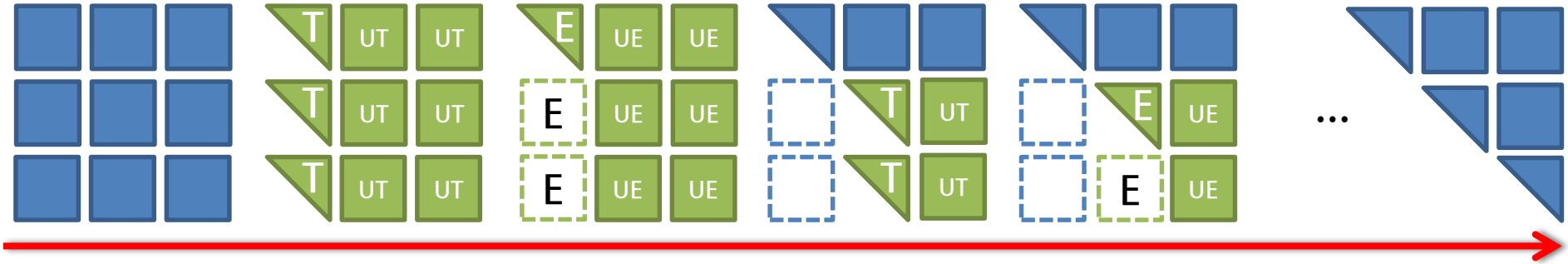
Performance Decreasing Factors

- Different **computation environments**
  - Core architecture, clock speed, memory bandwidth, …
  - Some jobs can be calculated faster on CPU
    - Jobs with low-parallelism

- Need of explicit **memory copy**
  - CPU and GPU cannot access each other's memory directly
  - Too many data to share → communication bottleneck → low utilization

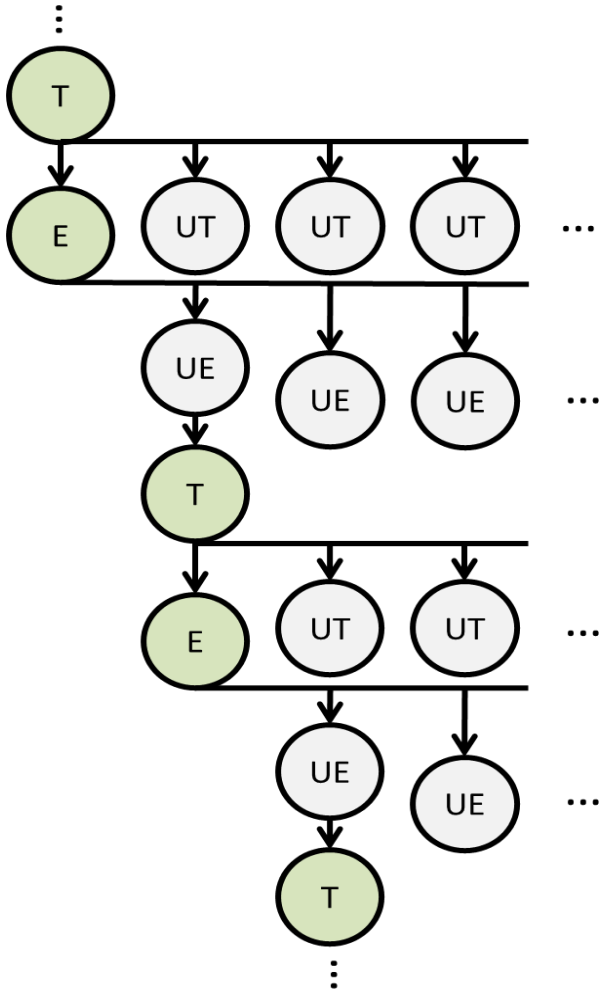
QR Decomposition

- QR Decomposition: A = QR
  - Q: Orthogonal matrix
  - R: Upper triangular matrix
- Tiled QR decomposition - for **parallelization**
  - Triangulation: Make upper triangle for a tile (T)
  - Elimination: Make zero matrix for T-ed tile from another T-ed tile (E)
  - Update-T: Update for right columns after T (uT)
  - Update-E: Update for right columns after E (uE)



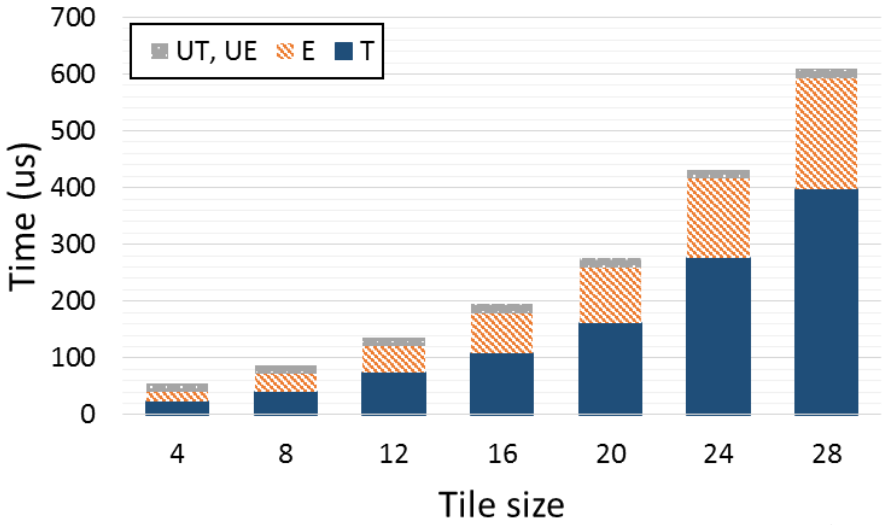
DAG of Tiled QR Decomposition

- Triangulation leads …
  - Elimination
  - Update for Triangulation
- Elimination leads …
  - Update for Elimination
- Update for Elimination leads …
  - Triangulation (next column)



Load Change within Each QR Step

- Calculation time
  - Two update processes are faster than Triangulation or Elimination
- Parallelism
  - Two update processes have much more tiles to be calculated
- → Separate Updates and Triangulation/Elimination on separated devices



〈Single tile operation on GTX680〉

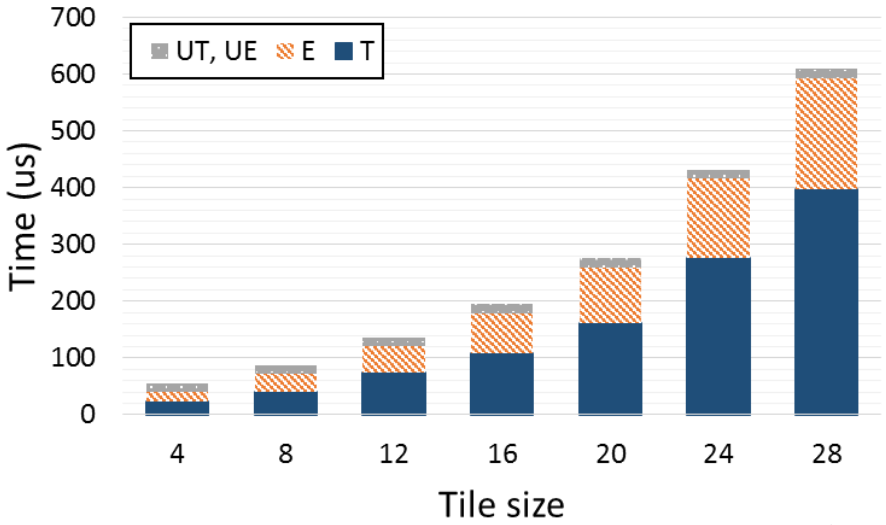
Step	Num. tiles
Triangulation	M
Elimination	M
Update for triangulation	$M \times (N-1)$
Update for elimination	$M \times (N-1)$

(Remaining part M by N)

〈The number of tiles to be operated〉

Heterogeneity of Computing Devices

- Heterogeneous environment
  - Different architecture, clock speed, …
- Triangulation and Elimination
  - Less tiles than Updates
  - More computing power for a tile
  - → Device's **speed**!
- Update processes
  - More tiles
  - Less computing power for a tile
  - → Device's **parallelism**!
- → Find appropriate device



〈Single tile operation on GTX680〉

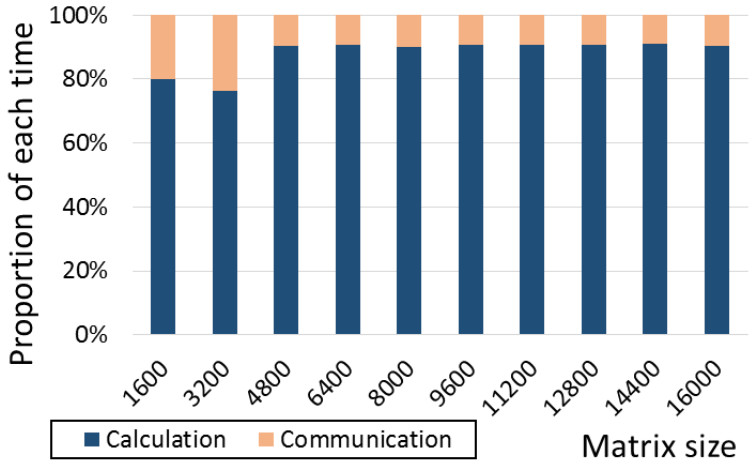
Step	Num. tiles
Triangulation	M
Elimination	M
Update for triangulation	$M \times (N-1)$
Update for elimination	$M \times (N-1)$

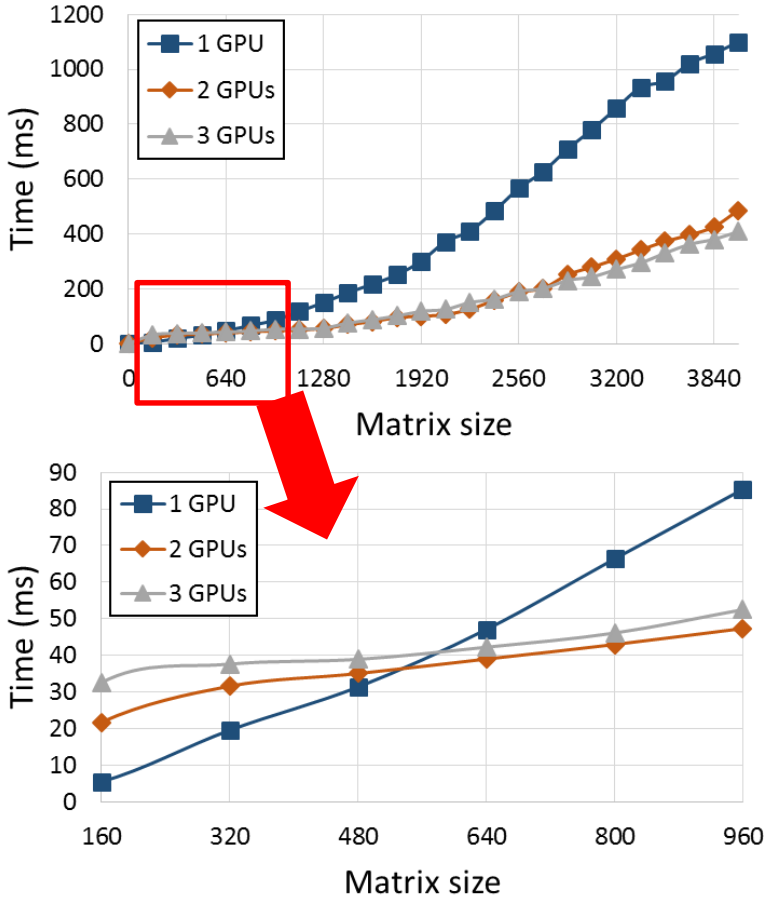
(Remaining part M by N)

〈The number of tiles to be operated〉

Effect of the Number of Devices

- More data transfer time if the number of devices increases
  - Trade-off between more parallel threads vs. comm. overhead
- → Find optimal number of devices for given matrix





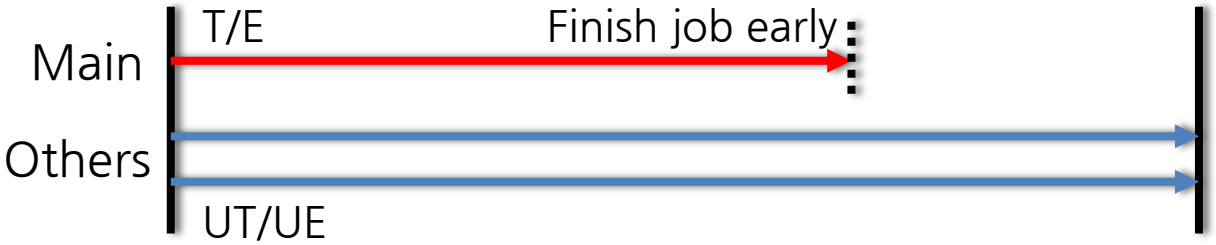
〈Total operation time〉

Contributions

- **Optimize** tile distribution and the tiled QR decomposition operation **mathematically**
  - Divided QR decomposition steps into appropriate computing devices
    - Depending on the processing properties
  - Optimize the number of devices that participate in the tiled QR decomposition
    - Depending on processing speed and communication cost
  - Tile distribution based on the parallelism of each device

Main Computing Device Selection

- Main Computing Device
  - Mainly executes the **triangulation** and **elimination** processes
- How to select
  - Can it finish its job before other's update processes?
    - Pre-processing → measure each device's calculation time
    - Multiply the number of tiles to be calculated
    - Determine whether a device can finish its job before others
    - From above, select a device that has less parallel cores
      - Since T/E have lower parallelism



The Number of Devices Selection (1)

- Find best number of devices
  - To optimize **trade-off** between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

$$T_{op}(p) = \max_{1 \le i \le p} \left[\begin{array}{c} \#tile_m \times (time_m(T) + time_m(E)) \\ +\#tile(m) \times (time_m(UT) + time_m(UE)) \\ or \\ \#tile(i) \times (time_i(UT) + time_i(UE)) \end{array} \right]$$

The Number of Devices Selection (1)

- Find best number of devices
  - To optimize trade-off between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

$$T_{op}(p) = \max_{1 \le i \le p} \left[\begin{array}{c} \#tile_m \times (time_m(T) + time_m(E)) \\ +\#tile(m) \times (time_m(UT) + time_m(UE)) \\ or \\ \#tile(i) \times (time_i(UT) + time_i(UE)) \end{array} \right]$$

The number of tiles, distributed to each device

Time taken for each step on each device

The Number of Devices Selection (1)

- Find best number of devices
  - To optimize trade-off between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

$$T_{op}(p) = \max_{1 \le i \le p} \left[\begin{array}{c} \#tile_m \times (time_m(T) + time_m(E)) \\ +\#tile(m) \times (time_m(UT) + time_m(UE)) \\ or \\ \#tile(i) \times (time_i(UT) + time_i(UE)) \end{array} \right]$$

Expected time for main computing device

The Number of Devices Selection (1)

- Find best number of devices
  - To optimize trade-off between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

$$T_{op}(p) = \max_{1 \le i \le p} \left[\begin{array}{c} \#tile_m \times (time_m(T) + time_m(E)) \\ +\#tile(m) \times (time_m(UT) + time_m(UE)) \\ or \\ \#tile(i) \times (time_i(UT) + time_i(UE)) \end{array} \right]$$

Expected time for other devices

The Number of Devices Selection (2)

- How to select (cont'd)
  - For all available devices, calculate expected communication time

$$T_{comm}(p) = \sum_{i=1}^{p} \left(3MT^2 \times size(element) \times \frac{1}{speed(m,p)} \right)$$
$$+(M-1)T^2 \times size(element) \times \frac{1}{speed(j,m)}$$

The Number of Devices Selection (2)

- How to select (cont'd)
  - For all available devices, calculate expected communication time

$$T_{comm}(p) = \sum_{i=1}^{p} \left(\begin{array}{l} 3MT^2 \times size(element) \times \frac{1}{speed(m,p)} \\ +(M-1)T^2 \times size(element) \times \frac{1}{speed(j,m)} \end{array} \right)$$

The number of tiles to be transferred

Time taken for each step on each device

Transfer speed

The Number of Devices Selection (2)

- How to select (cont'd)
  - For all available devices, calculate expected communication time

$$T_{comm}(p) = \sum_{i=1}^{p} \left(3MT^2 \times size(element) \times \frac{1}{speed(m,p)} \right) + (M-1)T^2 \times size(element) \times \frac{1}{speed(j,m)}$$

Expected time for Triangulation and Elimination

MT: Result Q matrices of Triangulation
2MT: Result Q matrices of Elimination

The Number of Devices Selection (2)

- How to select (cont'd)
  - For all available devices, calculate expected communication time

$$T_{comm}(p) = \sum_{i=1}^{p} \left(3MT^2 \times size(element) \times \frac{1}{speed(m,p)} \right)$$

$$+(M-1)T^2 \times size(element) \times \frac{1}{speed(j,m)}$$

Expected time for next column tiles

The Number of Devices Selection (2)

- How to select (cont'd)
  - For all available devices, calculate expected communication time

$$T_{comm}(p) = \sum_{i=1}^{p} \left(3MT^2 \times size(element) \times \frac{1}{speed(m,p)} \right) + (M-1)T^2 \times size(element) \times \frac{1}{speed(j,m)}$$

  - Find **p** which minimizes $T_{op}(p) + T_{comm}(p)$, $1 \leq p \leq N$

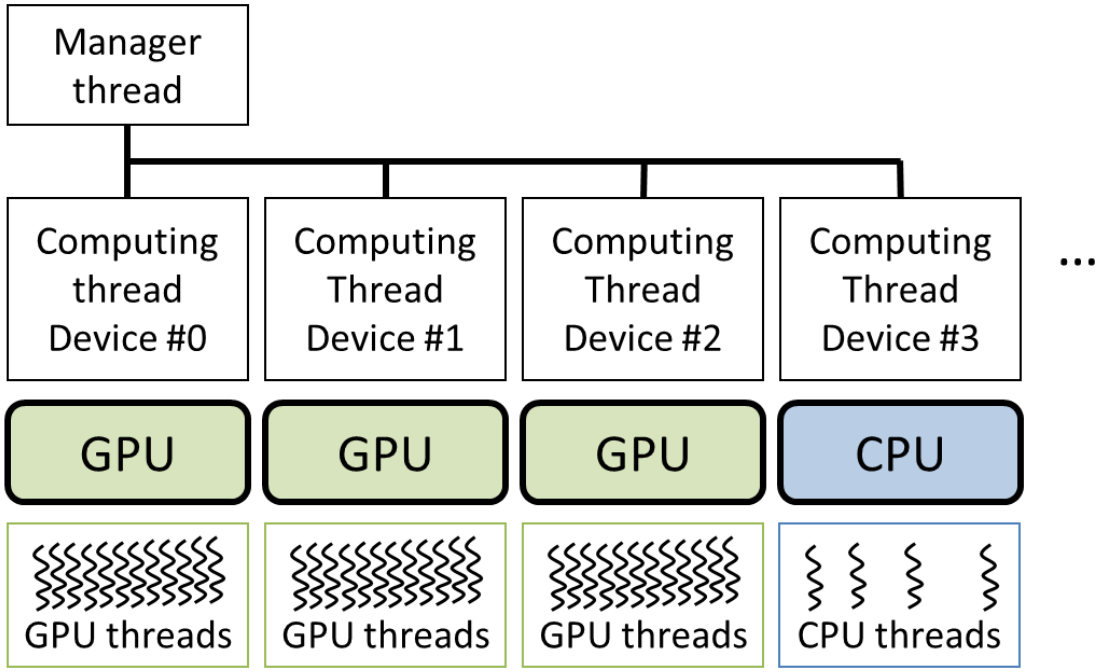
Tile Distribution

- Distribute tiles on each device
  - All devices should finish its job synchronously to maximize performance
- Load balancing based on distribution guide array
  - An array consists of device IDs
  - Find integer ratio of all devices, based on the number of tiles to be processes on fixed time
    - Device ID 0,1,2 and performance 3:2:1 → [0,1,2,0,1,0]
    - The count of each ID is proportional to the performance
  - Distribute each column tile

$$distribute(i) = guide\_array[i\%length(guide\_array)].$$

Implementation

- Manager thread
  - Select main computing device, decide the number of participating devices, distribute tiles, and migrate dependent data
- Computing thread
  - Do its own job
  - Have multiple slave threads for parallel operation



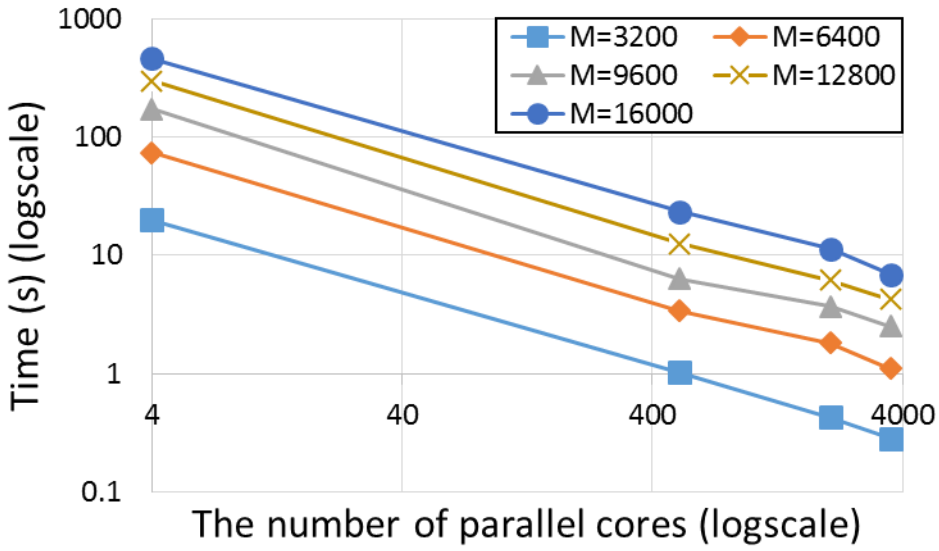
Evaluation Environment

- CPU
  - Intel i7-3820 (Quad core, 3.6GHz)
- Main Memory
  - 32GB
- GPU
  - Two GTX680 (1536 cores) + one GTX580 (512 cores)
- OS
  - Ubuntu 12.04, with Linux 3.2.0
- GPU driver version
  - 304.54
- CUDA version
  - 5.0

Scalability

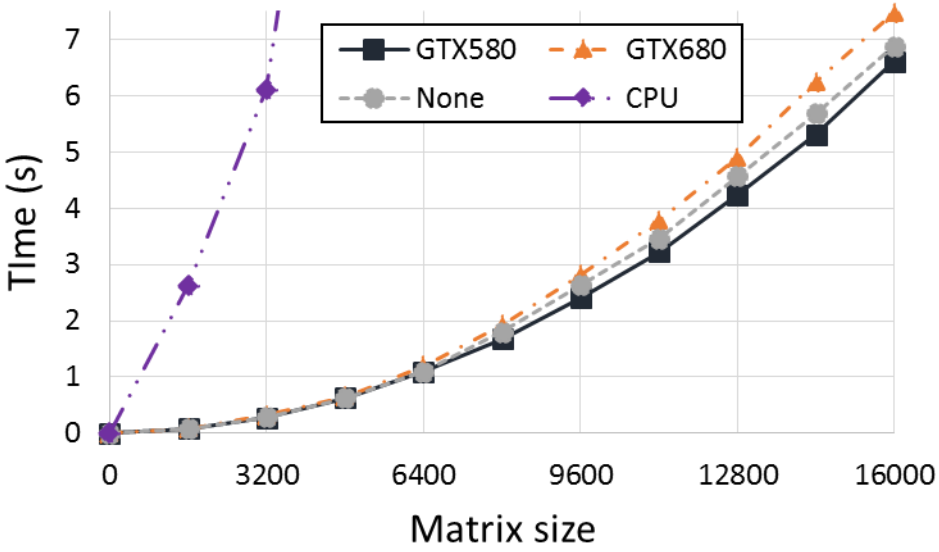
- Time taken for …
  - Only CPU: 4 cores
  - CPU+1GPU: 516 cores
  - CPU+2GPUs: 2,052 cores
  - CPU+3GPUs: 3,588 cores

Total operation time proportionally decreases



Effect of Main Computing Device Selection

- Total operation time, with changing the main computing device selection
  - With our algorithm: GTX580 was selected as main computing device
  - 13% speed-up with another GPU as main computing device
  - 5% speed-up without specific main computing device



Effect of The Number of Devices Selection

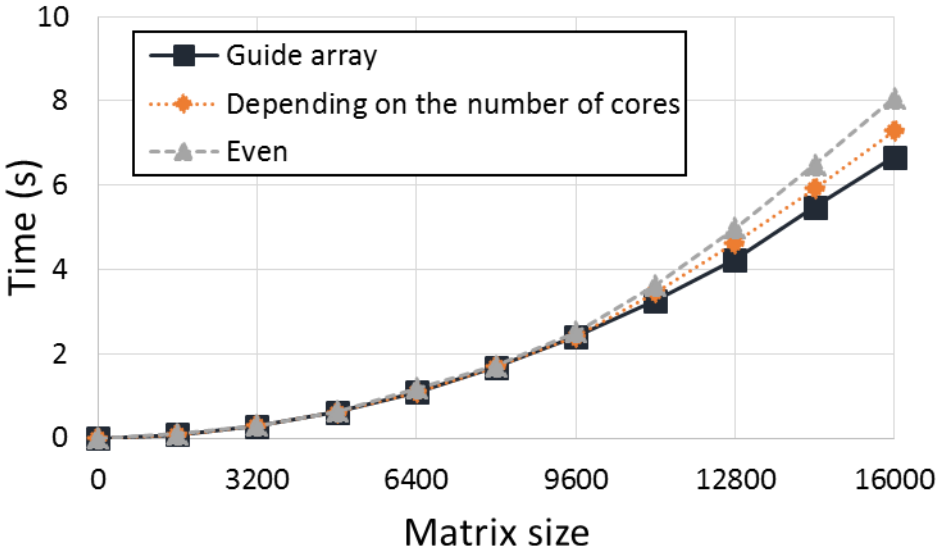
- Compare predicted optimal number and actual optimal number
- Our algorithm can find actual optimal number of devices

	Predicted			Actual		
Matrix size	1 G	2 G	3 G	1 G	2 G	3 G
160	**1.00**	3.73	5.27	**1.00**	4.00	5.99
320	**1.00**	2.77	3.82	**1.00**	1.62	1.93
480	**1.00**	1.40	2.11	**1.00**	1.12	1.24
640	1.02	**1.00**	1.43	1.21	**1.00**	1.08
800	1.03	**1.00**	1.34	1.54	**1.00**	1.07
960	1.03	**1.00**	1.29	1.80	**1.00**	1.11
1120	1.06	**1.00**	1.23	2.35	**1.00**	1.07
1280	1.09	**1.00**	1.19	2.71	**1.00**	1.02
1440	1.13	**1.00**	1.16	2.68	**1.00**	1.09
1600	1.15	**1.00**	1.13	2.71	**1.00**	1.09
1760	1.18	**1.00**	1.10	2.63	**1.00**	1.06
1920	1.20	**1.00**	1.08	3.00	**1.00**	1.18
2080	1.22	**1.00**	1.05	3.52	**1.00**	1.18
2240	1.23	**1.00**	1.04	3.27	**1.00**	1.19
2400	1.25	**1.00**	1.02	3.07	**1.00**	1.03
2560	1.26	**1.00**	1.01	3.02	**1.00**	1.01
2720	1.28	1.01	**1.00**	3.12	1.01	**1.00**
2880	1.31	1.02	**1.00**	3.07	1.09	**1.00**
3040	1.34	1.03	**1.00**	3.20	1.14	**1.00**
3200	1.36	1.04	**1.00**	3.18	1.14	**1.00**
3360	1.38	1.05	**1.00**	3.17	1.16	**1.00**
3520	1.40	1.07	**1.00**	2.90	1.13	**1.00**
3680	1.42	1.07	**1.00**	2.82	1.09	**1.00**
3840	1.44	1.08	**1.00**	2.78	1.12	**1.00**
4000	1.46	1.09	**1.00**	2.69	1.19	**1.00**

(*) Normalized value for smallest time

Effect of Tile Distribution

- Check the performance with Distribution Guide Array
- 21% faster than evenly distributed case
- 10% faster than distribution just based on the number of cores



Conclusion

- Summary
  - Mathematical optimization for tile QR decomposition
    - On CPU and GPU heterogeneous computing system
  - Select a specific device as the main computing device
    - Handles Triangulation and Elimination
  - The number of device optimization
  - Distribution based on distribution guide array
  - Algorithms can optimize the performance

- Further works
  - Considering very large matrix operation
    - Lack of memory problem will appear
  - Expand algorithms into other computing systems
    - Generalization

Thank YOU!

Any Question?